A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language

Wei Tan, Yushun Fan, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Automatic Web Service composition is gaining momentum as the potential *silver bullet* in Service Oriented Architecture. The need for interservice compatibility analysis and indirect composition has gone beyond what the existing service composition/verification technologies can handle. Given two services whose interface invocation constraints are described by a Web Services-Business Process Execution Language (WS-BPEL or BPEL), we analyze their compatibility and adopt mediation as a lightweight approach to make them compatible without changing their internal logic. We first transform a BPEL description into a service workflow net, which is a kind of colored Petri net (CPN). Based on this formalism, we analyze the compatibility of two services, and then devise an approach to check whether there exists any message mediator so that their composition does not violate the constraints imposed by either side. The method for mediator generation is finally proposed to assist the automatic composition of partially compatible services. Our approach is validated through a real-life case and further research directions are pointed out.**

***Note to Practitioners*—Web services are an emerging area for business process automation. This work presents a novel and fundamental framework to analyze and compose web services. It is based on CPNs and a newly proposed concept called Communicating Reachability Graph. Given any two services developed using Business Process Execution Language, the proposed method converts them into CPNs and analyzes their compatibility. Meanwhile, it finds out if a message mediator exists in case of partial compatibility. If so, it is synthesized. A real-life case is used to illustrate the feasibility of the proposed concepts and method. They can be readily used in industrial web service composition for business automation.**

***Index Terms*—Business process execution language (BPEL), mediation, Petri nets, web service composition.**

Manuscript received June 17, 2007; revised August 28, 2007. This paper was recommended for publication by Associate Editor S. Keerthi and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by the IBM Ph.D. Fellowship, the National Science Foundation of China under Grant 60674080, the National 973 Foundation of China under Grant 2006CB705407, and the Chang Jiang Scholar Program, PRC Ministry of Education.

W. Tan and Y. Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: tanwei@mails.tsinghua.edu.cn; fanyus@tsinghua.edu.cn).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA, and also with the School of Electro-Mechanical Engineering, Xidian University, Xi'an, 710071 China (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2008.916747

I. Introduction

With the emergence of Service Oriented Architecture (SOA), service composition is gaining momentum as the potential *silver bullet* for the seamless integration of heterogeneous computing resources, rapid deployment of new business capabilities, and enhanced reuse possibilities to a variety of legacy systems [1]. Existing service composition specification languages such as Web Service - Business Process Execution Language for Web Services (WS-BPEL or BPEL) [2], Web Service Choreography Interface (WSCI) [3], and Web Service Choreography Description Language (WS-CDL) [4] all provide mechanisms to *directly compose* two (or more) services by specifying that a message sent by one interface is received by the other (and *vice versa*). This kind of interface link is achieved using constructs such as *Partner Link* in BPEL, *Connect* in WSCI, and *Channel* in WS-CDL. Among these various specification languages, BPEL is becoming dominant because it has been proposed by OASIS as an industry standard and is supported by major software companies such as IBM, Oracle, and SAP. In this paper, we take BPEL as the language for describing the internal logic of web services.

Direct composition is made based on the assumptions that:

1) The incoming messages of one service are the exact ones provided by its partner(s); the outgoing messages of one service are the exact ones consumed by its partner(s).
2) Two services in composition consent to the message format and exchange sequence such that their composition process always executes in a logically correct way (e.g., terminating properly).

While existing specifications and approaches in service composition mostly concentrate on direct composition, it is observed that *partial compatibility* is common in real-life web service composition. Partial compatibility refers to the situation that two (or more) web services provide complementary functionality and could be linked together in principle; however, their interfaces and interaction patterns do not fit each other exactly. Hence, they cannot be directly composed. The problem of partial compatibility arises mainly because services have to interact with one another in the ways not necessarily foreseen when they are separately developed. Consequently, the assumptions made for direct composition may no longer hold.

As presented in Section II, if two services have mismatches in their interfaces and interaction patterns, they cannot be directly composed. Furthermore, even if they can be directly composed, we do not have confidence in the correctness of their composition (the meaning of correctness depends on specific requirements, but usually proper terminability, liveness, and bounded-

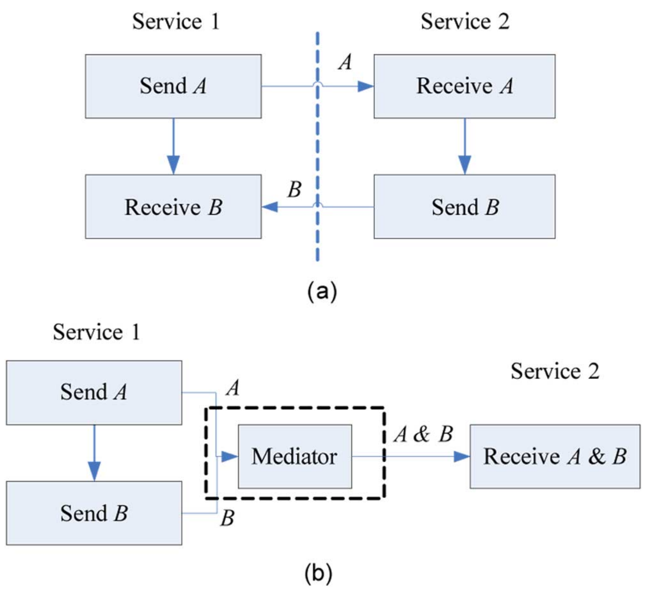


Fig. 1. (a) Direct composition. (b) Mediation-aided composition.

ness are essential ones among them [5]). However, the current research in web service composition mainly focuses on automatic composition methods [6]–[9] and formal verification of service composition [10]. Less attention is paid to the issue of partial compatibility.

Recently, the mediation approach is attracting more attention [11]–[15], because it is considered as an economic and labor-saving approach to address the challenge of partial compatibility in real-life web service composition. The basic idea of mediation-aided composition is similar to the concept of *adapter* to make two pieces of hardware compatible. Mediator wraps the various services such that they can appear homogeneous and are therefore easier to be integrated.

Fig. 1 illustrates direct and mediation-aided compositions. Messages A and B are exchanged between Services 1 and 2. In Fig. 1(a), Service 1 first sends A and then waits for B; Service 2 first waits for A and then sends B. Services 1 and 2 can be directly composed by adding links between "Send A" and "Receive A", "Receive B" and "Send B". In Fig. 1(b), Service 1 first sends A and then sends B; Service 2 waits for A and B to arrive simultaneously. Services 1 and 2 cannot be directly composed by simply adding links, because Service 1 has two interfaces, while Service 2 has only one. In this case, we can add a module (i.e., *mediator*) between them. Its function is to combine messages A and B coming from Service 1, and forward it to Service 2. With the aid of this mediator, Services 1 and 2 can be correctly composed.

Fig. 1(b) presents only a rather simple case of mediation-aided composition, and later we will see more powerful mediators that can intercept, store, transform, and forward messages between services.

In our work, we use colored Petri nets (CPNs) as an underlying formalism. This formal model provides not only a formalism to depict the internal logic and the message exchange behavior, but also rich analysis capability to support the solid verification of compatibility and mediation existence.

To the best of our knowledge, currently there is neither formal analysis on the existence of mediation nor methods to derive the mediator for two BPEL web services. Compared with the existing work, our contributions are the following.

1) In contrast with the existing approaches assuming a web service as a set of independent operations, our approach takes into account the conversational nature of web services. We take the *de facto* standard, i.e., BPEL, as our input, and define service workflow net (SWF-net) - a kind of CPNs, as a unified formalism to describe services, composition, and mediator.
2) We use a state-space based method to check the existence of mediation. We introduce the concept of Communicating Reachability Graph (CRG) whose function is to concurrently construct the reachability graph of two services, using data mapping (that needs to be provided as an input in our current solution) as the communication mechanism.
3) By using the concept of stubborn sets, we verify that the CRG contains all needed properties in mediation existence checking. Moreover, state-space exploration is sped up.
4) We propose the guidance to generate a mediator to glue two services if through CRG we can verify that mediation is possible.
5) We validate the approach through a real-life case.

The rest of this paper is organized as follows. Section II presents a motivating scenario for compatibility analysis and mediation-aided composition. Section III formally defines the concept of *service, composition, mediation*, and *mediation-aided composition*. Section IV introduces the method to check whether there is any mediator to glue two partially compatible services, and Section V introduces the method to generate the mediator. Section VI summarizes the related work and Section VII concludes the work and suggests future research directions.

II. Motivating Scenario

The motivating scenario in this paper comes from the composition of eBay and a third-party checkout (TPC) service [16]. It is the excerpt of a real bbusiness scenario, simple yet illustrative enough to explain the proposed concept, algorithm, and result.

eBay, an online auction and shopping service provider, allows a third party to handle a seller's checkout processes. When buyers on eBay finish shopping and want to check out, they are directed to a checkout system provided by a TPC service provider. The eBay and TPC services are to be composed to fulfill the requirement of the online shopping and checkout business. They provide complementary functionality, but since they are developed by different entities, they do not fit each other exactly.

Fig. 2 shows the BPEL of both eBay and TPC services. When a buyer wins an auction, the eBay service starts the checkout, and then invokes a TPC service by passing *OrderID, UserID*, and *SecretID*. Next, eBay receives invocation from TPC (*Receive FetchToken*) with *UserID* and *SecretID*, and replies with *Token* (*Token* is a required parameter in subsequent interaction). Finally, eBay receives invocation from TPC with *Token, OrderId*, and *UserID* (*Receive GetOrderData*), and replies with

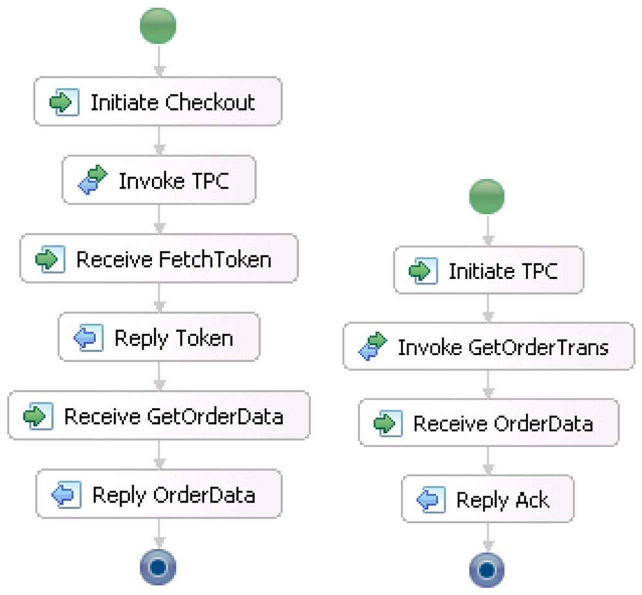


Fig. 2. BPEL of eBay and TPC services.

OrderData. On the TPC side, the TPC service initiates upon receiving *Order* and *PartnerID*. Then, it invokes *GetOrderTrans* with *OrderID* and *UserID*, and then waits for an asynchronous reply with *OrderData* from the online merchant (*Receive OrderData*). TPC returns an *Ack* after it receives *OrderData* (*Reply Ack*). With the order data retrieved, TPC shows to the user the items purchased and shipping address, and the buyer follows the TPC flow to complete the checkout process (this step is not relevant to the problem to be addressed. Hence, it is not shown in Fig. 2).

Obviously, these two services cannot be directly composed because of the following reasons.

1) The messages exchanged between their operations do not match exactly.
2) The numbers of their operations and input/output messages are not identical.

Besides, without formal analysis, the correctness of their composition cannot be guaranteed.

In the above scenario, there is no way to compose eBay and TPC services directly. So, the current direct composition approaches cannot be used here. To solve this problem, Section III gives the formalism of BPEL services, mediation, and composition, and these formalisms are the bases for the rigorous approach to be proposed in Section IV.

III. Formalism of BPEL Service, Mediation, and Composition

Using Petri nets to model BPEL processes is not a new idea. However, our formalism separates control flow with the message exchange, and provides a unified formalism to describe services, composition, and mediator. We also provide a new correctness criteria regarding service composition. These can be viewed as novel development in applying Petri nets to this area.

A. CPN Formalism for BPEL Process

To formally analyze the compatibility and mediation issue in BPEL services' composition, we first define a formal model. Then, we present how to transform a BPEL process to this model.

TABLE I
The Polarity of Places

Place	○	○→	→○	otherwise
Polarity(Ψ)	0	+1	-1	undefined

The definition of service workflow net is based on the concept of CPNs [17].

Definition 1. (Service Workflow Net, SWF-net): A Service Workflow Net is a CPN (P, T, F, Σ, C):

1) P is a finite set of places. $P = P_I \cup P_M$ and
   i) P_I is the set of internal places.
   ii) P_M is the set of message places. For $\forall p \in P_M, (\bullet p = \emptyset \wedge |p \bullet| = 1) \vee (p\bullet = \emptyset \wedge |\bullet p| = 1)$.
   iii) $P_I \cap P_M = \emptyset$.
2) T is a finite set of transitions, disjoint from P; $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation F, and (P_I, T, F) is a WF-net [5], i.e.,
   i) (P_I, T, F) has two special places: i and o. Place i is a source place: ${}^\bullet i = \emptyset$; Place o is a sink place: $o^\bullet = \emptyset$.
   ii) If we add a transition t to (P_I, T, F) which connects place o with i (i.e., ${}^\bullet t = \{o\}, t^\bullet = \{i\}$), then the resulting Petri net is strongly connected.
3) Σ is a finite set of color sets.
4) C is the color function, defined from P to Σ; $\forall p \in P_I, C(p) = e$. Color set $E = \{e\} \in \Sigma$ has only one possible value which stands for the control token in (P_I, T, F).

Remarks:

1) Internal places P_I represent the internal control logic, and message places P_M represent the messages exchanged between services.
2) (P_I, T, F) forms an ordinary WF-net whose definition is given in [5].
3) The internal places are tagged with color e that stands for the control token in (P_I, T, F).
4) The message places are tagged with the color sets that represent message types.

To differentiate input/output message places, we define *message polarity* for message places, denoted as Ψ

$$\forall p \in P_M$$
$$\Psi(p) = \left\{ \begin{array}{ll} +1; {}^\bullet p = \emptyset \wedge |p^\bullet| = 1, & \text{(incoming message)} \\ -1; p^\bullet = \emptyset \wedge |{}^\bullet p| = 1, & \text{(outgoing message)} \\ 0; p^\bullet = {}^\bullet p = \emptyset & \\ \text{undefined; otherwise.} & \end{array} \right\}$$

The idea of message polarity comes from [15], and it is notated, as shown in Table I.

⟨receive⟩, *⟨reply⟩*, *⟨invoke⟩*, *⟨sequence⟩*, *⟨if⟩*, *⟨pick⟩*, *⟨flow⟩*, *⟨while⟩*, *⟨repeatUntil⟩*, and *⟨link⟩* are key elements to describe the control logic of BPEL, in WS-BPEL specification 2.0. Since our work tackles the issue of service compatibility in control-logic aspect, it does not consider BPEL constructs such as *compensation, fault handler, assign, correlation set, link condition*, and *variables*.

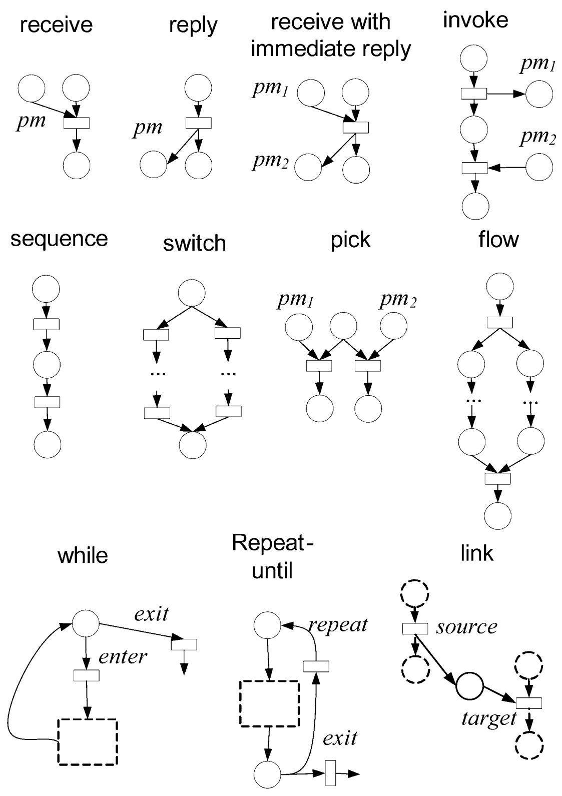


Fig. 3. Transforming BPEL into SWF-net.

Fig. 3 illustrates how to transform some BPEL elements into SWF-net ones. BPEL activities and internal control logic are modeled with internal places and transitions; the messages exchanged are modeled with message places. In Fig. 3, message places are explicitly denoted (pm, pm_1 and pm_2), and other places are all internal places.

B. Service Composition

Before defining service composition, we define the fusion of two CPNs.

Definition 2. (Fusion of CPNs): Given two CPNs $N_i = (P_i, T_i, F_i, \sum_i, C_i, M_{0i}), i = 1, 2$. If $P_C = P_1 \cap P_2 \neq \emptyset$. $\forall p \in P_C, C_1(p) = C_2(p), M_{01}(p) = M_{02}(p)$. Then, $N = (P, T, F, \sum, C, M_0)$ is the fusion of N_1 and N_2 via places, iff the following requirements are satisfied.

1) $P = P_1 \cup P_2$.
2) $T = T_1 \cup T_2, T_1 \cap T_2 = \emptyset$.
3) $F = F_1 \cup F_2$.
4) $\forall p \in P$, if $p \in P_1, C(p) = C_1(p)$; else $C(p) = C_2(p)$.
5) $\forall p \in P$, if $p \in P_1, M_0(p) = M_{01}(p)$; else $M_0(p) = M_{02}(p)$.

We claim that two CPNs N_1 and N_2 are *fusible* if they could be fused via a set of common places with identical color sets and marking, and in this case the fusion of N_1 and N_2 via common places P_C is denoted as $N_1 \oplus_{P_C} N_2$.

Remark: In order to do place fusion, we should first label the places that we want to merge in two nets with identical labels, and we should also guarantee that places which we do not want to merge are with different labels. P_C is the set of common places in CPNs, which have identical color set definitions and markings in two nets.

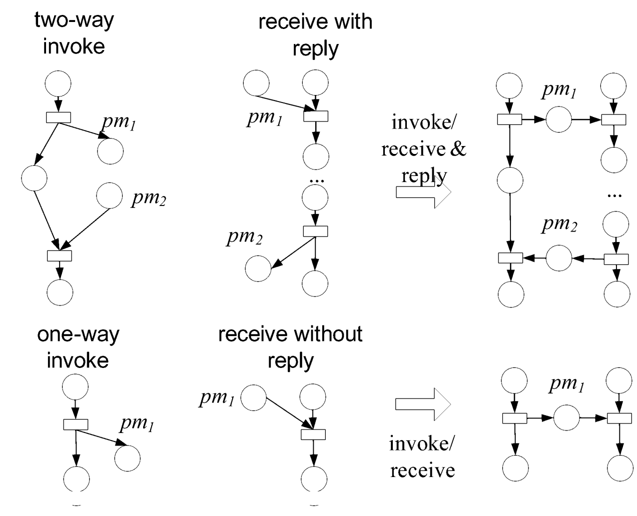


Fig. 4. Transform BPEL composition to SWF-net composition.

Definition 2 is based on the definition given in [18]. However, when we consider the problem of service composition, there are some specific requirements.

1) Only message places can be merged, and thus we should pay special attention when we define P_C.
2) When we merge two message places from two services, we require that, not only their color sets be identical, but also their polarity be contrary.

Given these requirements which are specific to service composition, we define the composition of two SWF-nets.

Definition 3. (Composition of SWF-Nets): Given two SWF-nets $N_i = (P_{Ii} \cup P_{Mi}, T_i, F_i, \Sigma_i, C_i), i = 1$ and 2. $N = (P_I \cup P_M, T, F, \Sigma, C)$ is the composition of N_1 and N_2 iff:

1) $P_{I1} \cap P_{I2} = \emptyset, P_I = P_{I1} \cup P_{I2}$.
2) $P_M = P_{M1} \cup P_{M2}, P_C = P_{M1} \cap P_{M2} \neq \emptyset$.
3) $\forall p \in P_C, p$ in N_1 and p in N_2 are with contrary polarity $(\Psi_1(p)\Psi_2(p) = -1)$. Ψ_1 and Ψ_2 are the polarity functions of N_1 and N_2, respectively.
4) $N = N_1 \oplus_{P_C} N_2$.

The composition of two SWF-nets N_1 and N_2 via common message places P_C is denoted as $N = N_1 \otimes_{P_C} N_2$. We claim that operator $\otimes_{P_C}$ can be extended to other CPNs as long as their places are classified into two disjoint categories, i.e., internal places (P_I) and message places (P_M). Later, we use this operator when we define composition via mediation.

Fig. 4 presents the method to transform BPEL composition to SWF-net composition. In Fig. 4, the composition of ⟨*invoke*⟩ and ⟨*receive*⟩/⟨*reply*⟩ is modeled with CPN composition.

C. Mediator and Mediation-Aided Service Composition

When one service provides functionality that another service requires, it is unlikely to directly compose the two services if they are not programmed to collaborate in advance. In our formalism, there does not exist a set P_C such that $N = N_1 \otimes_{P_C} N_2$.

We introduce the concept of *mediation* to deal with this difficult issue. Informally, *mediation* (*or mediator*) is a piece of code that sits between two services and compensates for the differences between their interfaces. Formally, we model a mediator as a CPN which has interfaces to the two services that want to be composed to collaborate. All messages exchanged by the two services go through the mediator.

Definition 4. (Mediation/Mediator): A mediator $\mathbf{M} = (P_m, T_m, F_m, \Sigma_m, C_m)$ is a place-bordered CPN. The polarity of the border places can be $0, -1$ or $+1$.

Remark: A place-bordered Petri net is a net in which 1) every transition has at least one input and one output places and 2) some places have input or output transitions but not both, which are called border places.

Definition 5. (Composition via mediator): Given two SWF-nets N_1, N_2 and a mediator $\mathbf{M}, N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$ is the composition of N_1 and N_2 via mediator $\mathbf{M}$.

Remark:

1) Intuitively, service composition is to synthesize two SWF-nets via message places with identical color set and contrary polarity.
2) If $P_{C1} \cap P_{C2} \neq \emptyset, \forall p \in P_{C1} \cap P_{C2}, p$ in N_1 and p in N_2 are with contrary polarity ($\Psi_1(p)\Psi_2(p) = -1$), and in $\mathbf{M}, \Psi_{\mathbf{M}}(p) = 0$.
3) $N_1 \otimes_{P_C} N_2$ is a special case of $N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$, where $P_C = P_{C1} = P_{C2}$, and $T_m = F_m = \emptyset$.

Now, we can give the definition of service compatibility.

Definition 6. (Compatibility of Two Services): Given SWF-nets N_1 and N_2, N is the composition of N_1 and N_2, i.e., $N = N_1 \otimes_{P_C} N_2$. $M_0 = M_{10} \times M_{20}, M_e = M_{1e} \times M_{2e}$. M_{i0} and M_{ie} are the initial and final markings of N_i, respectively.

N_1 is compatible with N_2 with respect to P_C iff the reachability graph of N is well-formed, i.e.,

1) $\forall M \in \mathbf{R}(N, M_0)$, there is a firing sequence σ and a marking M_σ s.t. $M[\sigma\rangle M_\sigma$ and $M_\sigma \geqslant M_e$.
2) Given $M \in \mathbf{R}(N, M_0)$ s.t. $M \geqslant M_e$, if $\exists p \in PM(p) > M_e(p)$, then $p \in P_{M1} \cup P_{M2}$.

In addition, two SWF-nets N_1 and N_2 are compatible with mediator $\mathbf{M}$, iff the reachability graph of $N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$ is well-formed.

The meaning of Definition 6 can be explained as follows.

1) Once the interaction begins, it will complete successfully.
2) When interaction completes, both services reach their ending states, and possibly there are remaining messages that are sent out by one service but not consumed by the other.

Weak soundness [19] requires a stronger condition than the above-defined compatibility does, and *soundness* [5] requires more than *weak soundness* does. This relaxation roots from the observation that services are more autonomous and loosely coupled. Hence, Definition 6 is more reasonable in stating what condition should be satisfied when two services are believed to be compatible.

Definitions 5 and 6 concern only the composition of two services. Our method can easily be extended to a multiservice composition scenario by stepwise composition and analysis. Hence, we focus our discussion on two-service composition thereafter.

IV. Mediator Existence Checking With Communicating Reachability Graph (CRG)

The solution approach is illustrated in Fig. 5. First, transform two BPEL services to be composed into SWF-nets. Then, verify whether they are directly composable. If not, request data mapping information. Next use data mapping to build CRG to verify whether there exists a mediator to glue them. If yes, generate it. In this section, we use the motivating scenario presented in Section II to explain our solution approach.

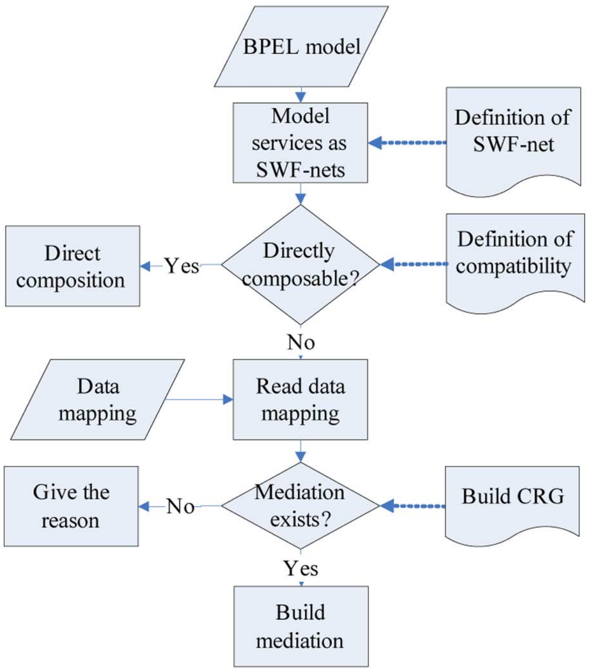


Fig. 5. The proposed solution approach.

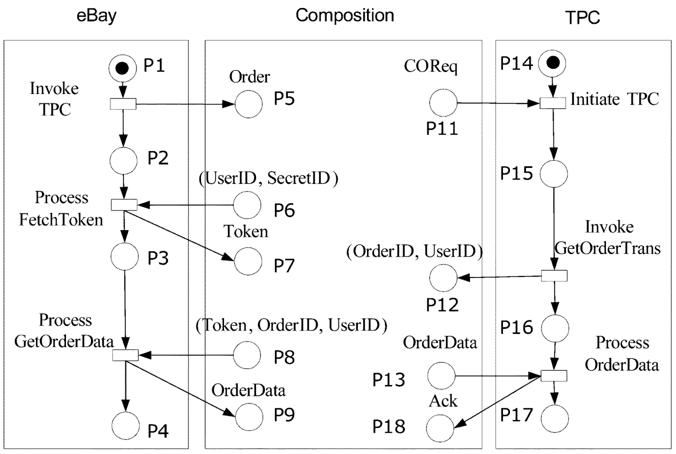


Fig. 6. Two SWF-nets: eBay and TPC.

A. *Transforming Abstract BPEL Process to SWF-Net*

The method to transform BPEL services to SWF-nets has been given in Section III. Fig. 6 depicts the result of transforming two BPEL services in Fig. 2 into SWF-nets. In Fig. 6, message *Order* is defined as the composition of *OrderId, UserID* and *SecretID*; and message *COReq* is defined as the composition of *Order*, and *PartnerId*. Transitions *Process FetchToken, Process GetOrderData*, and *Process OrderData* are all shorts for *receive* plus (immediate) *reply* in the original BPEL process. We make this simplification to obtain a more compact state space while preserving all the behavioral characteristics. In Fig. 6, we group the message places of two SWF-nets into a rectangle called *composition*.

B. Specifying Data Mapping

The functionality of data mapping is to define rules to relate (syntactically/semantically equivalent) elements of two messages such that two interfaces belonging to different services can be linked, and ultimately, services can be composed. In the BPEL specification, messages exchanged between web services are modeled as an aggregation of parts and/or elements. Therefore, data mapping can be at message level, part level or element level.

It is noted that data mapping is a very challenging problem in the area of data integration, semantic web, and SOA. This paper assumes that the data mapping relation between two services to be composed is specified by service composers who intend to compose them. This paper then focuses on how to detect the behavior mismatches between two services, and how to build a mediator to glue them (if possible). Semantic web service technology allows the semiautomatic and automatic annotation of web services, and therefore could be very helpful in the generation of data mapping relations between two services. Cardoso and Sheth [20] proposed an semantic-based approach to map data between the input and output messages of web services. Szomszor *et al.* [21] presented a method to harmonize syntactically incompatible service interfaces in web service composition, and the mediation of data format is based on mappings between XML schemas and OWL ontology. These works focus on the data semantics and transformation between web service interfaces, but our work focuses on the behavior analysis and mediation via data transformation. Therefore, on one hand, our work has different emphasis; on the other hand, these two approaches could be combined to provide a total solution to more adaptive and semantic-enhanced web service composition.

Data mapping I between N_1 and N_2 is a finite set of data mapping rules. Each rule is expressed in the form of ⟨*src, target, trans_flag*⟩, where *src* can be a message, part or element of a message whose place is with the polarity of -1 (i.e., an outgoing message place); *src* can also be a constant. *target* can be a message, a part, or element of a message whose place is with the polarity of $+1$ (i.e., an incoming one). *trans_flag* is a Boolean variable and the default value is *false*. It is set to be *false* if *target* can be mapped from *src* directly, and it is set to be *true* if *target* can be mapped from *src* with additional transformation.

The data mapping should have the following properties.

Completeness: Every element/part of a target message appears on the right-hand side of some mapping rule.

Accuracy: Every element/part of a target message appears in at most one right-hand side of a mapping rule.

In the eBay example, we have the data mapping shown in Table II. *eBay* and *TPC* stands for eBay and TPC services, respectively. Notations like *eBay.Order* stands for message *Order* of *eBay*. For complex messages made up of multiple parts/elements, we use brackets to aggregate the multiple parts/elements into one single message. For example, message *eBay.(UserID, SercretID)* stands for a message of eBay, and this message is made up of two elements, i.e., *UserID* and *SercretID*. In *eBay.(Token, OrderID, UserID), Token* stands for the element *Token* in message *(Token, OrderID, UserID)* of *eBay*. In row 3, "eBay" stands for a constant string whose content is *eBay*.

TABLE II
DATA MAPPING TABLE

Source	Target
eBay.Order	TPC.COReq.Order
"eBay"	TPC.COReq.PartnerId
eBay.Order.(UserID, SercretID)	eBay.(UserID, SercretID)
TPC. (OrderID, UserID)	eBay.(Token, OrderID, UserID).(OrderID, UserID)
eBay.Token	eBay. (Token, OrderID, UserID).Token
eBay.OrderData	TPC.OrderData

Our approach assumes that the data mapping between two services is accurate. It is easy to verify that the data mapping in Table II is complete and accurate.

C. Mediator Existence Checking

In order to check whether there exists mediation to glue two partially compatible services, we introduce the concept of CRG. Its basic idea is to construct the reachability graph of two services concurrently, using data mapping as the communication mechanism. That is, when the source data is ready, their target should be informed.

Given two SWF-nets N_1, N_2, and a data mapping I, their CRG is a directed graph $\mathbf{G}(N_1, N_2, I) = (V, E)$, where $V \subseteq M_1 \times M_2$. There are two kinds of edges in CRG. One is operation edge, i.e.,

$$\begin{aligned} E_O = \{\langle m, t, m'\rangle | m, m' \in V \text{ and } m \xrightarrow{t} m', \\ \text{if } t \in T_1, m' = m_1' \times m_2; \text{ else if } t \in T_2, m' \\ = m_1 \times m_2'\}. \end{aligned}$$

The other is mediation edge, i.e.,

$$E_M = \{\langle m, t_M(m), m'\rangle | m, m' \in V \text{ and } m \xrightarrow{t_M(m)} m'\}.$$

A mediation edge is constructed to act as the communication mechanism between two services. The meaning of $t_M(m)$ is given in Algorithm 1.

Algorithm 1. (Method to Construct CRG)

Input: SWF-nets $N_i = (P_{Ii} \cup P_{Mi}, T_i, F_i, \Sigma_i, C_i), i = 1, 2$ and data mapping I.
The data mapping table used here is similar to Table II, except that a new column named flag is added.

Flag	Source	Target
	SrcMsg_1.Element_1	TargetMsg_1.Element_1
	…	…
	SrcMsg.Element_*n*	TargetMsg.Element_*n*

Output: $\mathbf{G}(N_1, N_2, I) = (V, E)$

1) Initialize $(V, E) = (\{m_0 = m_{10} \times m_{20}\}, \emptyset)$; m_0 is untagged.

2) While there are untagged nodes in V do

```
2.1) Select an untagged node $m \in V$ and tag it; ($m = m_1 \times m_2$)
2.2) For each enabled transition, $t_i \in T_i$, at $m = m_1 \times m_2$, do
    2.2.1) Compute $m'$ such that $m \xrightarrow{t_1} m'$; if $i = 1, m' = m'_1 \times m_2$; else if $i = 2, m' = m_1 \times m'_2$.
    2.2.2) If there exists $m$" such that $m'' \xrightarrow{\sigma} m', m'' \leqslant m' \wedge m'' \neq m'$, and $\exists p \in P \wedge p \notin P_{M1} \cup P_{M2}$ s.t. $m''(p) < m(p)$, then the algorithm fails and exists; (the unboundedness condition of CRG has been detected).
    2.2.3) If there is no $m'' \in V$ such that $m'' = m'$ then $V = V \cup \{m'\}$; ($m'$ is untagged).
    2.2.4) $E = E \cup \{(m, t, m')\}$.
    2.2.5) If $t_i^\bullet \cap P_{Mi} = p_m$ (i.e., new tokens are fed into message places $p_m$ with the firing of $t_i$).
        a) Flag all the rows whose source message is $C(p_m)$ in the data mapping table.
        b) Compute $m''$ such that $m' \xrightarrow{t_M(m')} m''$. $m''$ is derived from $m'$ through virtual transition $t_M(m')$ by i) unmarking $p_m$ if in all data mapping rules whose source message is $C(p_m)$, the target messages are satisfied; and ii) for each newly satisfied target message in step a, mark the place attached with that message.
           (A target message is *satisfied* iff in the data mapping table, all the rows it resides in are flagged).
        c) If $m'' \neq m'$ : tag $m'$; $V = V \cup \{m''\}$; $E = E \cup \{(m', t_M(m'), m'')\}$. ($m''$ is untagged).
3) The algorithm succeeds with $\mathbf{G}(N_1, N_2, I)$ obtained.
```

Given eBay service N_1, TPC service N_2 in Fig. 6, and the data mapping I in Table II, we can derive $\mathbf{G}(N_1, N_2, I)$ according to Algorithm 1, as Fig. 7 shows. The operation edges are denoted with solid lines, and the names of operation transitions are labeled on the lines. The mediation edges are denoted with dashed lines, and the data obtained by mediation are labeled on the dashed lines.

We will prove that if $\mathbf{G}(N_1, N_2, I)$ satisfies certain property, there exists a mediator to glue N_1 and N_2, and in the next section, we give the method to generate it. We have the following theorem.

Theorem 1: Given accurate data mapping I, and two SWF-nets N_1 and N_2, there exists a mediator $\mathbf{M}$ with respect to I, and N_1 and N_2 can be composed via $\mathbf{M}$ iff $\mathbf{G}(N_1, N_2, I)$ is well-formed, i.e.,

1) $\forall M \in \mathbf{R}(M_0)$, there is an edge sequence σ and a marking M_σ s.t. $M[\sigma\rangle M_\sigma$ and $M_\sigma \geqslant M_e$.

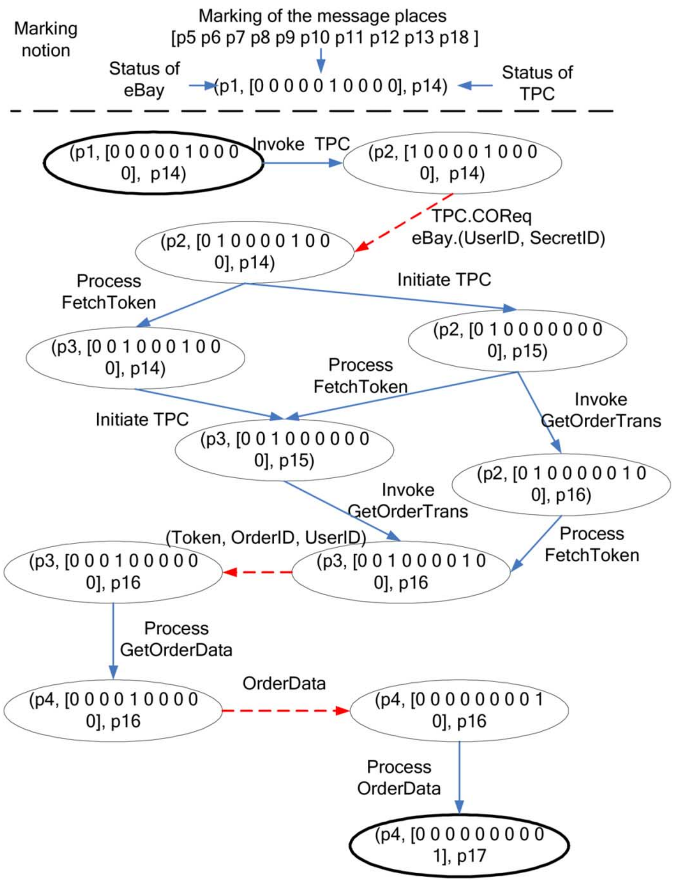

Marking notion
Marking of the message places [p5 p6 p7 p8 p9 p10 p11 p12 p13 p18]
Status of eBay
(p1, [0 0 0 0 0 1 0 0 0 0], p14)
Status of TPC
(p1, [0 0 0 0 0 1 0 0 0 0], p14)
Invoke TPC
(p2, [1 0 0 0 0 1 0 0 0 0], p14)
TPC.COReq
eBay.(UserID, SecretID)
(p2, [0 1 0 0 0 0 1 0 0 0], p14)
Process FetchToken
Initiate TPC
(p3, [0 0 1 0 0 0 1 0 0 0], p14)
(p2, [0 1 0 0 0 0 0 0 0 0], p15)
Process FetchToken
Invoke GetOrderTrans
Initiate TPC
(p3, [0 0 1 0 0 0 0 0 0 0], p15)
(p2, [0 1 0 0 0 0 0 1 0 0], p16)
Invoke GetOrderTrans
(Token, OrderID, UserID)
Process FetchToken
(p3, [0 0 0 1 0 0 0 0 0 0], p16)
(p3, [0 0 1 0 0 0 0 1 0 0], p16)
Process GetOrderData
OrderData
(p4, [0 0 0 0 1 0 0 0 0 0], p16)
(p4, [0 0 0 0 0 0 0 0 1 0], p16)
Process OrderData
(p4, [0 0 0 0 0 0 0 0 0 1], p17)

Fig. 7. The CRG of eBay and TPC service.

2) Given $M \in \mathbf{R}(M_0)$ s.t. $M \geqslant M_e$, if $\exists p \in P, M(p) > M_e(p)$, then $p \in P_{M1} \cup P_{M2}$.

Its proof is given in the appendix. The proof shows that the CRG contains all needed properties in mediator existence checking. Moreover, in our method to build CRG, state-space exploration is sped up, because mediation transitions are selected as stubborn sets. Using stubborn sets, CRG is generated as a reduced state-space of the composed services, eliminating some intermediate states that are not relevant to compatibility verification.

We can easily verify that $\mathbf{G}(N_1, N_2, I)$ in Fig. 7 is well-formed. Therefore, we claim that a mediator exists to glue N_1 and N_2.

Fig. 8(a) shows two services with no mediator to glue them. Service X expects to receive message B, and then sends message A in the next step. Service Y expects to receive message A, and then sends message B in the next step. With data mapping $< X.A, Y.A >, < X.B, Y.B >$, the CRG [Fig. 8(c)] is not well-formed. Therefore, X and Y cannot be composed with the aid of mediation. Fig. 8(b) shows the composition of services X and Y with given data mapping, and the net obtained contains a deadlock in fact.

V. Mediator Generation Approach

A. *Types of Mediation*

Here, we first introduce the classification of mediators.

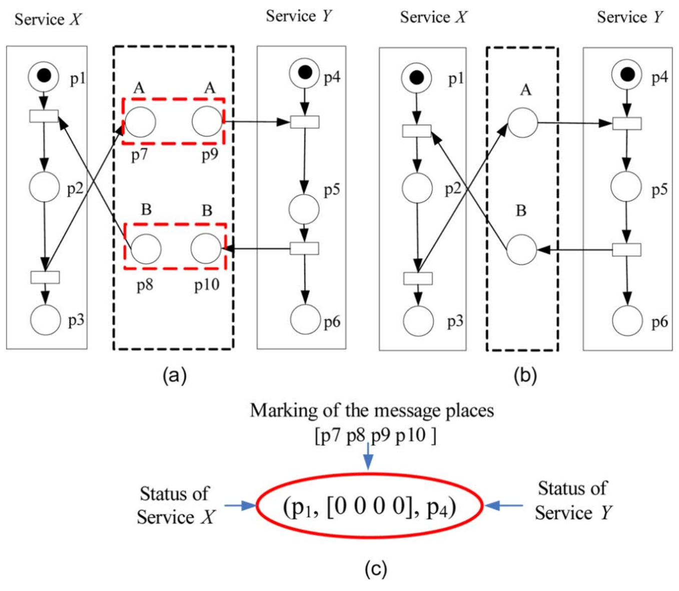


Fig. 8. (a) An example in which no mediator exists. (b) The composed service. (c) The CRG.

1) Store/Forward Mediator: A store/forward mediator is the simplest kind. It stores the incoming message and forwards it to the receiver when needed.

2) Transformation Mediator: A transformation mediator transforms the incoming message and forwards it to the receiver when needed.

3) Split Mediator: A split mediator replicates the source message (or part/element of it) into multiple copies.

4) Merge Mediator: Corresponding to a split mediator, a merge mediator collects the multiple source messages/parts/elements, and then combines them into one single target message.

The CPN models for these types of mediators are given in Fig. 9.

Given the different types of mediation, in Lemma 1, we prove that the value of *trans_flag* does not influence the existence of mediators. Hence, it is ignored in the mediator existence checking algorithm.

Lemma 1: The value of *trans_flag* does not influence the existence of mediator.

Proof: For a data mapping rule ⟨*src, target, trans_flag*⟩, the mediator generated is shown in Fig. 10. The mediator with transformation can be reduced to the mediator w/o transformation, preserving liveness, safeness, and boundedness (this kind of reduction is called Fusion of Series Places in [22]). From Theorem 1, we see that mediator existence only relates to the liveness and safeness property. Thus, the value of *trans_flag* does not influence the existence of mediators.

The mediator existence checking method assumes the value of *trans_flag* to be false in each data mapping rule. Owing to Lemma 1, it can be extended to circumstances where *trans_flag* is *true* or *false*.

B. Guided Mediator Generation

In Section IV, we use CRG to check the existence of mediator between two services. If it is well-formed, this section gives the method to build the mediator between message places of N_1 and N_2.

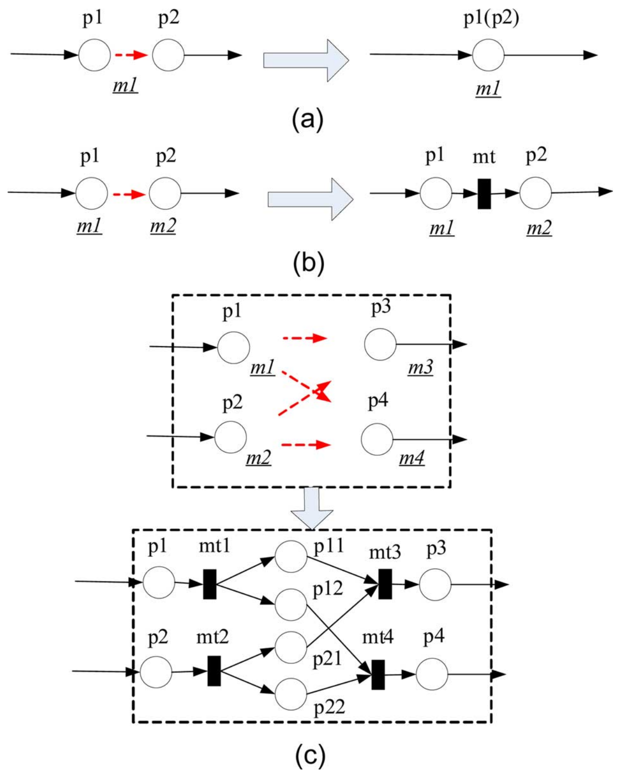


Fig. 9. (a) Store/forward mediator. (b) Transformation mediator. (c) Split/merge mediator.

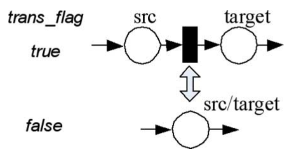


Fig. 10. Transformation mediator and store/forward mediator.

Recall the concept of data mapping, and its example in Table II, we define *reference count* for each message m_s in data mapping I. Message m_1 is referred by message m_2, if m_1 and m_2 appear in one row in the data mapping table. For example, in Table II, the first row represents data mapping rule ⟨eBay.Order, TPC.COReq.Order⟩, we say that message *Order* of eBay is referred by message *COReq* of TPC, and *vice versa*. The reference count for message m is the number of data mapping rules whose source message is m.

For example, in Table II, the reference count of message *eBay.Token* is 1, since it is only referred by *eBay*. (*Token, OrderID, UserID*); the reference count of message *eBay.Order* is 2, since it is referred by *TPC.COReq* and *eBay* (*UserID, SecretID*.)

While reference count can be easily derived from a simple observation on the data mapping table, we further define *active reference count* (Δ) for each source message m in data mapping I. $\Delta(m)$ is the number of data mapping rules whose source message is m, and the target of this data mapping is consumed in CRG. For example, the reference count of *eBay.OrderData* is 1; if in Fig. 7 there is always a token left in p_{13} in terminal marking M_e, the active reference count of *eBay.OrderData* is 0, because it means that although message *eBay.OrderData* is referred by one data mapping rule, the target message of this mapping is never consumed. A data mapping rule is *active* iff the target message is consumed in at least one terminal marking M_e.

Based on the concept of data mapping and active reference count, we can build a mediator between two services. Algorithm 2 gives the method to generate it. Its basic idea is illustrated in Fig. 9, and its procedure is explained as follows.

For a message mapping where $\Delta(src) = \Delta(\text{target}) = 1$, use a *store/forward mediator*. In Fig. 9(a), if message m_1 is sent by p_1 and needed by p_2, we simply merge p_1 and p_2 to form a *store/forward mediator*.

For message mapping with transformation, use a *transformation mediator*. In Fig. 9(b), if m_1 is sent by p_1 and m_2 is needed by p_2, and m_2 can be transformed from m_1; we add a mediation transition *mt* between p_1 and p_2 to form a *transformation mediator*.

For message *src* with $\Delta(src) > 1$, use a *split mediator*. For message *target* with $\Delta(target) > 1$, use a *merge mediator*.

In Fig. 9(c), if m_1 is sent by p_1 and referred by m_3 and m_4; m_2 is sent by p_2 and also referred by m_3 and m_4. We add split transitions *mt*$_1$ and *mt*$_2$ after p_1 and p_2, merge transitions *mt*$_3$ and *mt*$_4$ before p_3 and p_4, and we add p_{11}, p_{12}, p_{21}, and p_{22} to connect these mediation transitions.

Algorithm 2 (Generation of a Mediator)

```
Input: SWF-nets $N_1$ and $N_2$, data mapping $I$
Output: $\mathbf{M} = (P_m, T_m, F_m, \Sigma_m, C_m)$
1) $P_m = P_{M1} \cup P_{M2}, T_m = F_m = \emptyset$.
2) For each outgoing message place $p_i \in P_{M1} \cup P_{M2}$
   s.t. $\Psi(p_i) = -1$.
   2.1) If $k_i = \Delta(C(p_i)) > 1$
        2.1.1) Add a mediation transition $t_i$ in $\mathbf{M}$:
               $T_m = T_m \cup \{t_i\}$, add places
               $\{p_{i1}, p_{i2}, \ldots, p_{ik_i}\}$ in $\mathbf{M}$ :
               $P_m = P_m \cup \{p_{i1}, p_{i2}, \ldots, p_{ik_i}\}$.
        2.1.2) $F_m = F_m \cup \{(p_i, t_i), (t_i, p_{i1}),$
               $(t_i, p_{i2}), \ldots, (t_i, p_{ik_i})\}$.
3) For each incoming message place $p_j \in P_{M1} \cup P_{M2}$
   s.t. $\Psi(p_j) = +1$.
   3.1) If $k_j = \Delta(C(p_j)) > 1$
        3.1.1) Add a mediation transition $t_j$ in $\mathbf{M}$ :
               $T_m = T_m \cup \{t_j\}$, add places
               $\{p_{j1}, p_{j2}, \ldots, p_{jk_j}\}$ in $\mathbf{M}$ :
               $P_m = P_m \cup \{p_{j1}, p_{j2}, \ldots, p_{jk_j}\}$.
        3.1.2) $F_m = F_m \cup \{(t_j, p_j), (p_{j1}, t_j),$
               $(p_{j2}, t_j), \ldots, (p_{jk_j}, t_j)\}$.
4) For each active data mapping rule without transformation,
   and the source and target message places are $p_i$
   and $p_j$, respectively: $\mathrm{Merge}(p_i^\bullet, {}^\bullet p_j)$.
5) For each active data mapping rule with transformation,
   and the source and target message places are $p_i$
   and $p_j$, respectively, $\mathrm{AddTransformation}(p_i^\bullet, {}^\bullet p_j)$.
6) Return $\mathbf{M} = (P_m, T_m, F_m, \Sigma_m, C_m)$.
```

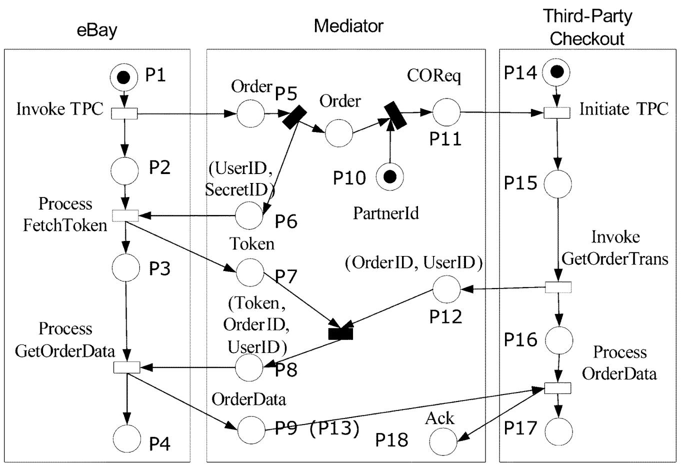


Fig. 11. Mediation-aided service composition.

In Algorithm 2, two functions, i.e., $\mathrm{Merge}(p_i^\bullet, {}^\bullet p_j)$ and $\mathrm{AddTransformation}(p_i^\bullet, {}^\bullet p_j)$ are used. $\mathrm{Merge}(p_i^\bullet, {}^\bullet p_j)$ is to merge a place $p_{ik} \in (p_i^\bullet)^\bullet$ with a place $p_{jk} \in {}^\bullet({}^\bullet p_j)$, if $p_i^\bullet$ and/or ${}^\bullet p_j$ is null, use p_i and/or p_j instead of p_{i_k} and p_{j_k}. $\mathrm{AddTransformation}(p_i^\bullet, {}^\bullet p_j)$ is to add a transition t_{ikj} between a place $p_{ik} \in (p_i^\bullet)^\bullet$ and a place $p_{jk} \in {}^\bullet({}^\bullet p_j)$ (that is, $T_m = T_m \cup \{t_{ikj}\}, F_m = F_m \cup \{(p_{ik}, t_{ikj}), (t_{ikj}, p_{jk})\}$), if $p_i^\bullet$ and/or ${}^\bullet p_j$ is null, use p_i and/or p_j instead of p_{i_k} and p_{j_k}. Note that if $|p_i^\bullet| > 1$ or $|{}^\bullet p_j| > 1$, choose a different k each time these two functions are invoked, so that each place in $p_i^\bullet$ will merge with a different place in ${}^\bullet p_j$, and each newly added transition will have different source and target places in $p_i^\bullet$ and ${}^\bullet p_j$.

With the proposed method, a mediator between eBay and TPC is generated, as shown in Fig. 11. The mediation transitions are denoted with black rectangles to differentiate them from operation transitions belonging to eBay and TPC services. With the generated mediator $\mathbf{M}$, the reachability graph of $N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$ is well-formed, i.e., eBay and TPC services is compatible with the aid of mediator $\mathbf{M}$.

For mediator development/deployment, we use IBM WebSphere Integration Developer [23] (WID) and Websphere Process Server [24] (WPS). A program takes the BPEL services defined in WID as input, transform the BPEL services into SWF-nets, and analyze their compatibility. If the two BPEL services are compatible, a mediator Petri net is generated and transformed back into a mediator BPEL service. The mediator BPEL service consists of such activities as *receive, assign*, and *invoke* and the links among them. Once all source messages are ready, the target message is generated and the corresponding interface is invoked. In our BPEL run-time environment WPS, each *porttype* is equipped with a message queue. Thus, the mediator could simply forward the target message whenever all the source messages are ready, being unaware of when the target messages will be consumed. The mediator BPEL assembles the two original BPEL services and makes them run smoothly with each other. Otherwise, if the two BPEL services are not compatible, our program can point out where the in-compatibility is, and give instructions to correct the in-compatibility (e.g., by adding data mapping or changing a message sequence).

VI. Related Work

This section gives an overview of the related work.

A. Web Service Composition

Industry and academia have different concentrations in web service composition. Industrial community focuses on composition languages, e.g., BPEL, WS-CDL, OWL-S [25], and the build-time/run-time environment to support them [8]. However, the existing composition specifications only support direct composition, which seems to be straightforward and often too rigid. Academic community focuses on automatic composition [8] through AI planning [7], constraint solving [6], and other approaches and the verification of service composition [10], [26] using Petri nets, process algebra, and automata as the formalism. The work concentrates on the conceptual level, making reasonable simplification on some nontrivial details (for example, the partial compatibility issue).

B. Business Process Integration

Compatibility problems also exist when two or more business processes are to be integrated [27]. Compared with traditional business processes, web services are autonomous, loosely coupled, and defined in a standard format. Hence, mediation can be used as a preferable approach than modifying internal structure when gluing partially compatible services.

C. Web Service Configuration

Briefly, there are two methods to make partially compatible services work smoothly with each other, i.e., mediation [13] and configuration [28]. Configuration is a heavy-weight approach that changes the original service within some predefined variable points to make it work smoothly with other services. Casati and Shan [28] propose a configuration-based approach that enables the dynamic and adaptive composition of e-services.

D. Petri Net Model of BPEL Processes

There are some studies on the modeling and analysis of a *single* BPEL processes using Petri nets. [29]–[31]. Hinz [31] presents a Petri net model for BPEL, and this model covers the exceptional behavior (e.g., faults, events, compensation). Lohmann [29] uses Petri nets to decide the controllability of a service, i.e., the existence of a partner process, and compute its operating guideline. Ouyang *et al.* [30] present a comprehensive and rigorously defined mapping of BPEL constructs into Petri net structures. These studies provide good formalisms for the analysis of a single BPEL process. Other work focuses on the analysis of the *direct* composition of two or more BPEL services [32]–[34]. Compared with the work in [32]–[34], we clearly move one step forward to tackle the issue of *in-direct* composition.

E. Component/Web Service Mediation

Compared with configuration, mediation is relatively a lightweight approach. The idea of mediation-aided service composition is stimulated by the related work in software engineering area. For example, *generative programming* [35] is a style of computer programming that uses automated source code creation through generic classes, prototypes, templates, aspects, and code generators to improve the productivity of programmers.

Yellin *et al.* [15] propose a method to augment object-oriented component interfaces with *protocols* that include sequencing constraints. The author defines *adaptors* that can be used to bridge the differences between components, and presents the method to automatically generate adaptors. Our work is stimulated by [15], but significantly different from [15] in the following aspects. First, the automata-based approach has limitations in modeling the concurrent behavior of web services (e.g., the *flow* construct in BPEL), while Petri nets are good at modeling such behavior. Second, the automata-based approach focuses on a message exchange sequence, but ignores the different exchange styles (e.g., more subtle behavior such as one-way invoke and two-way invoke, receive with/without reply, cannot be modeled by automata), while Petri nets can describe them well (as seen in Section III). Third, in the automata model in [15], all the state transitions are triggered by external messages. Thus, the internal nondeterminism cannot be modeled (e.g., the *if* construct in BPEL), while our SWF-net can model internal as well as external nondeterminism. Last, the automata-based approach can only model the synchronous semantics of composition, while Petri net-based one can model both synchronous and asynchronous semantics. This work uses asynchronous semantics that is realistic in industrial service composition.

The work in [11] and [13] provides various scenarios of mismatch patterns in web service composition. However, a comprehensive classification is missing. Some scenarios are presented with the corresponding solutions. However, these solutions seem elementary (presented with natural language), and based on proprietary models. Hence, they can hardly be used in practice.

Based on their previous work [11], Kongdenfha *et al.* [14] propose the use of an aspect-oriented programming (AOP) approach to weave adaptation solutions into the partially compatible services. The recent work in [12] proposes a method to automatically generate a mediator between two services. Nezhad *et al.* [36] also propose an automata-based method to model the protocol of service interaction and to identify mismatches automatically. They use a schema matchmaking technique to handle the issue of message mapping, and also propose some heuristic methods for deadlock resolution between two services.

A comparative summary of previous efforts in this area is given in Table III. The columns of the table correspond to the following criteria, where Y means yes and N means no.

TABLE III
A COMPARISON OF RELATED WORK ON SERVICE COMPOSITION/ANALYSIS

	MO	**AC**	**CM**	**MC**	**AM**	**CP**
Ouyang et al. [30]	PN	Y	N	N	N	N
Hamadi [34]	PN	N	Y	N	N	N
Kongdenfha [14]	N/A	N	Y	Y	N	N
Brogi [12]	YAWL	Y	Y	Y	N	N
Nezhad [36]	FSM	N	Y	Y	Y	N
Our approach	CPN	N	Y	Y	N	Y

- **MO** indicates the formal model used: FSM for finite-state machines, PN for Petri nets, CPN for colored Petri nets, and YAWL for Yet Another Workflow Language.
- **AC** indicates whether the formalism provides a representation of advanced BPEL constructs like event handler, fault handler, scope, etc.
- **CM** indicates whether a formalism considers the composition of multiple services.
- **MC** indicates whether mediation code generation method is given.
- **AM** indicates whether a method for automatic message mapping between services is given.
- **CP** indicates whether correctness proof for a given method is given.

VII. CONCLUSION

When two web services provide complementary functionality and could be linked together in principle, but their interfaces and interaction patterns do not fit each other exactly, they cannot be directly composed. The challenge here is to analyze interservice compatibility and automatically compose services with the minimum engineering cost. In this paper, we address this challenge with the help of a motivating scenario on the composition of eBay and a TPC service.

To provide a rigorous approach to analyze the compatibility and check whether there exists any mediator, we first transform BPEL services into service workflow nets which is a kind of CPNs. Based on this model, we analyze the compatibility of two services, and devise an approach to check whether there exists any message mediator so that their composition violates no constraints imposed by either side. Later the method for mediator generation is proposed to compose two partially compatible services.

We have developed a prototype system to validate our approach. In this system, BPEL processes are transformed to CPNs, and CRGs are built to check the compatibility. If two nets are compatible, mediation skeletons are given, and if not, detailed instructions for modification are provided. The recently developed elementary siphon concepts can be potentially used to overcome this complexity problem [38]–[42].

Future work includes applying the proposed method to more real-life cases. Reachability-based methods are generally computationally expensive. Although we have adopted a stubborn-set-based approach to generate a reduced state-space, the proposed methods still need to be further tested on more complicated cases.

At the same time, the next step is to add automatic data mapping functionality into our system, using the semantic-based approach. In this paper, we utilize product-specific property to facilitate mediator service generation, so another to-do task is to further investigate the mediator code generation method in more general case.

APPENDIX
PROOF OF THEOREM 1

The Proof of Theorem 1 needs the concept of stubborn set [37]. Intuitively, a stubborn set contains a set of transitions, and remains to be *stubborn* when a transition outside it fires. This property infers that in marking M, if $t \in T_S, t_1, t_2, \ldots, t_n \notin T_S$ and $M[t_1t_2\ldots t_n\rangle M_n[t\rangle M'_n$, then there is a marking M' such that $M[t\rangle M'[t_1t_2\ldots t_n\rangle M'_n$.

Lemma 2: [37, Th. 1.23] The ordinary state-space is a labeled directed graph (W, E), while the reduced state-space generated by stubborn sets is a labeled directed graph $(\underline{W}, \underline{E})$, then

If $s \in W$ and s is a terminal state, then $s \in \underline{W}$ and s is a terminal state of $\underline{W}$.

If s is a terminal state of $\underline{W}$, then $s \in W$ and s is a terminal state of W. $\square$

Lemma 3: [37, Th. 1.24] There is an infinite occurrence sequence in the reduced state-space if and only if there is an infinite occurrence sequence in the ordinary state-space. $\square$

Theorem 1 claims that the method to generate CRG is equivalent to generate reduced state space using the set of mediation transitions as stubborn sets. Since we are only interested in terminal states in the state-space, the reduced state-space preserves all relevant properties. That is to say, CRG is well-formed iff the two services can be composed with the help of a mediator.

Proof of Theorem 1: $\Rightarrow$ We are to prove that if $\mathbf{G}(N_1, N_2, I)$ is well-formed, a mediator $\mathbf{M}$ which conforms to I exists. In Section V, we give the method to derive a mediator based on the information we get from (a well-formed) CRG, and we denote the mediation as $\mathbf{M}(\mathbf{M} = (P_m, T_m, F_m))$.

When we build the reachability graph of $N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$ (denoted as $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$), if we use the stubborn set defined in (1), we'll obtain a reduced reachability graph of $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$, denoted as $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$

$$\mathbf{S}(m) = \begin{cases} \{t_m\}, \text{when } \exists t_m \in T_m, t_m \text{ is enabled at } m \\ T_1 \cup T_2 \cup T_m, \text{otherwise} \end{cases}. \tag{1}$$

The meaning of function $\mathbf{S}$ is explained as follows. When there is mediation transition enabled, choose it as the single enabled transition in the stubborn set at marking m. (If there are multiple mediation transitions enabled, choose any one of them as the single enabled transition in the stubborn set.) When there is no mediation transition enabled, the stubborn set equals to the set of all transitions in $N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2$. By this means, we obtain a subgraph of $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$.

According to Lemmas 2 and 3, $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ contains all terminal markings and one infinite path if there is one in $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$. Since the verification of well-formedness only involves the terminal markings, $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ preserve all the information we require. On the other hand, it is easy to see $\mathbf{G}(N_1, N_2, I)$ is equivalent to $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ with respect to marking $M\prime (M\prime = M_1 \times M_2)$. So, if $\mathbf{G}(N_1, N_2, I)$ is well-formed, $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ is also well-formed, and ultimately, $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ is well-formed. Therefore, we conclude that $\mathbf{M}$ serves as the mediation to make N_1 compatible with N_2.

Remark: $\Gamma_R = (V, E),\ V \subseteq M_1 \times M_2 \times M_{\mathbf{M}}$; $\mathbf{G} = (V', E'),\ V' \subseteq M_1 \times M_2$. $M_1,\ M_2$ and $M_{\mathbf{M}}$ are markings of N_1, N_2 and mediator $\mathbf{M}$, respectively. $\mathbf{G}$ and Γ_R are *equivalent*, means that there is a surjective mapping φ between Γ_R and $\mathbf{G}$, such that $\forall v \in V$ and $\forall v' \in V', \varphi(v) = v'$ if $\prod_{M_1 \times M_2}(v) = v'$. $(v_1, v_2) \in E \Rightarrow (\varphi(v_1), \varphi(v_2)) \in E' \vee \varphi(v_1) = \varphi(v_2)$. If in Γ_R there is $v_1[t_1\rangle v_2[t_2\rangle v_3 \ldots [t_k\rangle v_{k+1}$, then $\varphi(v_1) = v_1', \varphi(v_2) = \varphi(v_3) = \ldots = \varphi(v_{k+1}) = v_{k+1}'$. $(t_1, t_2, \ldots, t_k$ are mediation transitions in Γ_R. $\prod_{M_1 \times M_2}$ is the projection function that projects the marking at $M_1 \times M_2 \times M_{\mathbf{M}}$ to $M_1 \times M_2$.)

$\Leftarrow$ We are to prove that if a mediation $\mathbf{M}$ which conforms to I exists, $\mathbf{G}(N_1, N_2, I)$ is well-formed. As we have mentioned, if a mediation $\mathbf{M}$ which conforms to I exists, $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ is well-formed, and since $\Gamma_R(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$ is a subgraph of $\Gamma(N_1 \otimes_{P_{C1}} \mathbf{M} \otimes_{P_{C2}} N_2)$, so it is also well-formed, and ultimately $\mathbf{G}(N_1, N_2, I)$ is well-formed. □

Acknowledgment

W. Tan would like to thank J. Zhu and F. Rao from the IBM China Research Laboratory, for their suggestions in motivating the scenarios and improving the quality of this paper. The authors also thank the anonymous reviewers for their valuable comments.

References

[1] S. Staab, W. van der Aalst, V. R. Benjamins, A. Sheth, J. A. Miller, C. Bussler, A. Maedche, D. Fensel, and D. Gannon, "Web services: Been there, done that?," *IEEE Intell. Syst.*, vol. 18, pp. 72–85, 2003.

[2] OASIS, "Web Services Business Process Execution Language Version 2.0," 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html

[3] W3C, "Web Service Choreography Interface (WSCI) 1.0," 2002. [Online]. Available: http://www.w3.org/TR/wsci/

[4] W3C, "Web Services Choreography Description Language Version 1.0," 2004. [Online]. Available: http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217

[5] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Syst. Comput.*, vol. 8, pp. 21–66, 1998.

[6] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven web service composition in METEOR-S," in *Proc. IEEE Int. Conf. Services Comput.*, 2004, pp. 23–30.

[7] M. Carman, L. Serafini, and P. Traverso, "Web service composition as planning," in *Proc. ICAPS 2003 Workshop on Planning for Web Services*, 2003.

[8] N. Milanovic and M. Malek, "Current solutions for web service composition," *IEEE Internet Comput.*, vol. 8, pp. 51–59, 2004.

[9] B. Srivastava and J. Koehler, "Web service composition-current solutions and open problems," in *Proc. ICAPS 2003 Workshop on Planning for Web Services*, 2003, pp. 28–35.

[10] R. Hull and J. Su, "Tools for composite web services: A short overview," *ACM SIGMOD Record*, vol. 34, pp. 86–95, 2005.

[11] B. Benatallah, F. Casati, D. Grigori, H. Motahari-Nezhad, and F. Toumani, "Developing adapters for web services integration," in *Proc. Int. Conf. Adv. Inf. Syst. Eng. (CAiSE)*, 2005.

[12] A. Brogi and R. Popescu, "Automated generation of BPEL adapters," in *Proc. Int. Conf. Service Oriented Comput. (ICSOC)*, Chicago, IL, 2006.

[13] D. Fensel and C. Bussler, "The web service modeling framework WSMF," *Electron. Commerce Res. Appl.*, vol. 1, pp. 113–137, 2002.

[14] W. Kongdenfha, R. Saint-Paul, B. Benatallah, and F. Casati, "An aspect-oriented framework for service adaptation," in *Proc. Int. Conf. Service Oriented Compu. (ICSOC)*, Chicago, IL, 2006.

[15] D. M. Yellin and R. E. Strom, "Protocol specifications and component adaptors," *ACM Trans. Programming Languages and Syst.*, vol. 19, pp. 292–333, Mar. 1997.

[16] eBay, "Third Party Checkout," 2006. [Online]. Available: http://developer.ebay.com/DevZone/XML/docs/WebHelp/Checkout-Third_Party_Checkout.html

[17] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. London, U.K.: Springer-Verlag, 1995, vol. 2.

[18] C. Girault and R. Valk, *Petri Nets for Systems Engineering*. Berlin, Germany: Springer-Verlag, 2003.

[19] A. Martens, "On compatibility of web services," *Petri Net Newsletter*, vol. 65, pp. 12–20, Oct. 2003.

[20] J. Cardoso and A. Sheth, "Semantic e-workflow composition," *J. Intell. Inf. Syst.*, vol. 21, pp. 191–225, Nov. 2003.

[21] M. Szomszor, T. R. Payne, and L. Moreau, "Automated syntactic mediation for web service integration," in *Proc. Int. Conf. Web Services*, 2006, pp. 127–136.

[22] B. Hruz and M. C. Zhou, *Modeling and Control of Discrete Event Dynamic Systems*. London, U.K.: Springer.

[23] "IBM Websphere Integration Developer." [Online]. Available: http://www-306.ibm.com/software/integration/wid/

[24] "IBM Websphere Process Server." [Online]. Available: http://www-306.ibm.com/software/integration/wps/

[25] W3C, "OWL-S: Semantic Markup for Web Services," 2004. [Online]. Available: www.w3.org/Submission/OWL-S/

[26] S. Deng, Z. Wu, M. Zhou, Y. Li, and J. Wu, "Modeling service compatibility with pi-calculus for choreography," in *Proc. 25th Int. Conf. Conceptual Modeling (ER2006)*, Tucson, AZ, 2006, pp. 26–39.

[27] W. M. P. van der Aalst, "Loosely coupled interorganizational workflows: Modeling and analyzing workflows crossing organizational boundaries," *Inf. Manage.*, vol. 37, pp. 67–75, 2000.

[28] F. Casati and M. C. Shan, "Dynamic and adaptive composition of e-services," *Inf. Syst.*, vol. 26, pp. 143–163, 2001.

[29] N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg, "Analyzing interacting BPEL processes," in *Proc. Int. Conf. Business Process Management*, 2006, pp. 17–32.

[30] C. Ouyang, E. Verbeek, W. M. P. V. D. Aalst, S. Breutel, M. Dumas, and A. H. M. t. Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," Queensland Univ. Technol., Brisbane, Australia, Tech. Rep. 2174, 2006.

[31] S. Hinz, K. Schmidt, and C. Stahl, "Transforming BPEL to Petri nets," in *Proc. Int. Conf. Business Process Manage.*, 2005, pp. 220–235.

[32] A. Martens, "Analyzing web service based business processes," in *Proc. 8th Int. Conf. Fundamental Approaches to Softw. Eng.*, 2005, pp. 19–33.

[33] A. Martens, S. Moser, A. Gerhardt, and K. Funk, "Analyzing compatibility of BPEL processes," in *Proc. Int. Conf. Internet and Web Applications and Services/Advanced Int. Conf. Telecommun. (AICT-ICIW'06)*, 2006, pp. 147–147.

[34] R. Hamadi and B. Benatallah, "A petri net-based model for web service composition," in *Proc. 14th Australasian Database Conf. (ADC2003)*, 2003, pp. 191–200.

[35] K. Czarnecki and U. Eisenecker, *Generative Programming: Methods, Tools, and Applications*. Reading, MA: Addison-Wesley, 2000.

[36] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "SemiAutomated adaptation of service interactions," in *Proc. 16th Int. World Wide Web Conf. (WWW2007)*, 2007, pp. 993–1002.

[37] A. Valmari, "Stubborn sets for reduced state space generation," in *Advances in Petri Nets 1990, LNCS 483*, G. Rozenberg, Ed. New York: Springer, 1991, pp. 491–515.

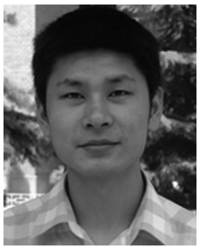
[38] Z. W. Li and M. C. Zhou, "Elementary siphons of Petri nets and their applications to deadlock prevention in flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 1, pp. 38–51, Jan. 2004.

[39] Z. W. Li and M. C. Zhou, "Clarifications on the definitions of elementary siphons in Petri nets," *IEEE Trans. Syst., Man, Cybern.: Part A*, vol. 36, no. 6, pp. 1227–1229, Nov. 2006.

[40] Z. W. Li and M. C. Zhou, "Two-stage method to design liveness-enforcing Petri net supervisor for FMS," *IEEE Trans. Ind. Inf.*, vol. 2, no. 4, pp. 313–325, Nov. 2006.

[41] Z. W. Li, H. S. Hu, and A. R. Wang, "Design of liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 4, pp. 517–526, Jul. 2007.

[42] Z. W. Li and M. C. Zhou, "Control of elementary and dependent siphons in Petri nets and their application," *IEEE Trans. Syst., Man, Cybern.: Part A*, vol. 38, no. 1, pp. 133–148, Jan. 2008.



Wei Tan received the B.S. and Ph.D. degrees in control theory and engineering from Tsinghua University, Beijing, China, in 2002 and 2008, respectively.

He is currently a Research Professional Associate at the Computation Institute, University of Chicago and Argonne National Lab, IL. His research interests include business process management, service oriented architecture, scientific workflow, and Petri net. From January to July 2007, he was a Visiting Researcher at the IBM T. J. Watson Research Center, NY.

Yushun Fan received the B.S. degree in automatic control from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1984, and the M.S. and Ph.D. degrees in control theory and application from Tsinghua University, Beijing, in 1987 and 1990, respectively.

He is currently a Professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. He authored ten books in enterprise modeling, workflow technology, intelligent agent, and object oriented complex system analysis, computer-integrated manufacturing, respectively, and published more than 300 research papers in journals and conferences. His research interest includes enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and workshop management and control.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, in 1990.

He joined the New Jersey Institute of Technology (NJIT), Newark, NJ, in 1990, and is currently a Professor of Electrical and Computer Engineering. His research interests are in computer-integrated systems, Petri nets, wireless ad hoc and sensor networks, system security, semiconductor manufacturing, and embedded control. He has over 250 publications including 6 books, 110 journal papers, and 16 book-chapters. His recent book is *Modeling and Control of Discrete Event Dynamic Systems*, Springer, 2007, coauthored with B. Hruz.

Dr. Zhou is a Life Member of the Chinese Association for Science and Technology-USA and served as its President in 1999. He was the recipient of NSF's Research Initiation Award, CIM University-LEAD Award by the Society of Manufacturing Engineers, the Perlis Research Award by NJIT, the Humboldt Research Award for U.S. Senior Scientists, and Leadership Award by the Chinese Association for Science and Technology, USA. He served as Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION from 1997 to 2000, and the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING from 2004 to 2007. He is currently Managing Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: PART C, Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: PART A, and Editor-in-Chief of the *International Journal of Intelligent Control and Systems*. He was General Co-Chair of the 2003 IEEE International Conference on System, Man, and Cybernetics, and General Chair of the 2006 IEEE International Conference on Networking, Sensors, and Control. He is serving as General Chair of the IEEE Conference on Automation Science and Engineering, Washington D.C., August 2008. He is Chair of the Semiconductor Manufacturing Automation Technical Committee of the IEEE Robotics and Automation Society and Co-Chair of the Enterprise Information Systems Technical Committee of the IEEE Systems, Man, and Cybernetics Society.